Teaching Large Language Models to Self-Debug

Year: 2023 | *Citations: 887* | *Authors: Xinyun Chen, Maxwell Lin, Nathanael Schärli*

Abstract

Large language models (LLMs) have achieved impressive performance on code generation. However, for complex programming tasks, generating the correct solution in one go becomes challenging, thus some prior works have designed program repair approaches to improve code generation performance. In this work, we propose Self-Debugging, which teaches a large language model to debug its predicted program via few-shot demonstrations. In particular, we demonstrate that Self-Debugging can teach the large language model to perform rubber duck debugging; i.e., without any human feedback on the code correctness or error messages, the model is able to identify its mistakes by investigating the execution results and explaining the generated code in natural language. Self-Debugging achieves the state-of-the-art performance on several code generation benchmarks, including the Spider dataset for text-to-SQL generation, TransCoder for C++-to-Python translation, and MBPP for text-to-Python generation. On the Spider benchmark where there are no unit tests to verify the correctness of predictions, Self-Debugging with code explanation consistently improves the baseline by 2-3%, and improves the prediction accuracy on problems of the hardest level by 9%. On TransCoder and MBPP where unit tests are available, Self-Debugging improves the baseline accuracy by up to 12%. Meanwhile, by leveraging feedback messages and reusing failed predictions, Self-Debugging notably improves sample efficiency, and can match or outperform baseline models that generate more than 10x candidate programs.